A single machine scheduling problem with two-dimensional vector packing constraints

A single machine scheduling problem with bin packing constraints

Jean-Charles BILLAUT [1]
[1] Université François-Rabelais Tours, CNRS
LI EA 6300, OC ERL-CNRS 6305
64 avenue Jean Portalis, 37200 Tours, France
jean-charles.billaut@univ-tours.fr

Federico DELLA CROCE [2]
D.A.I., Politecnico di Torino, Italy
federico.dellacroce@polito.it

Andrea GROSSO [3]
D.I., Universitá di Torino, Italy
grosso@di.unito.it

May 6, 2014

Abstract: In this paper we consider a scheduling problem where jobs consume a perishable resource stored in vials. It leads to a new scheduling problem, with two-dimensional jobs, one dimension for the duration and one dimension for the consumption. Jobs have to be finished before a given due date, and the objective is to schedule the jobs on a single machine so that the maximum lateness does not exceed a given treshhold and the number of vials required for processing all the jobs is minimized. We propose a two-step approach embedding a Recovering Beam Search algorithm to get a good-quality initial solution reachable in short time and a more time consuming matheuristic algorithm. Computational experiments are performed on the benchmark instances available for the two-constraint bin packing problem integrated with additional due dates to take into account the maximum lateness constraints. The computational results show very good performances of the proposed approach that remains effective also on the original two-constraint bin packing instances without due dates where 7 new bounds are obtained.

Keywords: scheduling, two-constraint bin packing, recovering beam search, matheuristic.

1 Introduction

Scheduling problems are well studied in the literature since the 50's and several books and recent review papers show the wide variety of problems considered today [Brucker, 2007, Pinedo, 2012, Ruiz and Vázquez-Rodrguez, 2010, Hartman and Briskorn, 2010]. In the same way, bin packing problems have received a great attention in the literature and some survey papers are dedicated to these problems [Lodi et al., 2002, De Carvalho, 2002]. In this paper, we introduce a new category of scheduling problems where bin packing constraints have to be considered together with the scheduling problem.

The origin of the problem comes from the production of chemotherapy drugs for cancer treatment by intravenous injection. In [Mazier et al., 2010], the authors describe this particular production environment and present a resolution method for a static or dynamic environment. In this production environment, the jobs to perform are called '*preparations*' and the raw materials are called *monoclonal antibodies* ('*products*' in the following). The monocolonal antibodies bind to specific cancer cells and induce an immunological response against the target cell. These products can be stored in vials for a long time before use, under specific storage temperatures and conditions. For some antibodies, freezing at $-20°$C or $-80°$C in small aliquots is the optimal storage condition. However, once a vial is opened or once the active agent has been mixed with a solute, it must be used before a given time limit, in order to keep intact the properties of the anticancer active agents. The maximum delay of use after opening depends on the agent and may vary between several hours to several days. In the mean time, the product has to be stored in a freezer or in a fridge for temperature and darkness reasons. If the time limit of use is exceeded, the monoclonal antibodies have lost their properties and they have to be destroyed according to a specific process. In other words, the time period between the starting time of the first preparation using a product in a vial and the completion time of the last preparation assigned to the same vial cannot be greater than the life time of the product. Furthermore, it is clear that the total consumption associated to one vial cannot be greater than the total volume of the vial. We have here the two constraints of the bin packing problem.

The cost of these products is very important. In Tours, the UBCO production center (described in [Mazier et al., 2010]) produces around 150 preparations per day. A preparation has an average cost of 400 euros, but it can reach 15,000 euros in certain cases. It is clear that the saving of these products may have an economic impact, absolutely not negligible. In the problem that we consider,

one objective function is related to the waste of products, that we want to minimize. Notice that in such a context, the deadline for the use of the raw material becomes a variable of the problem, which is directly related to the production scheduling decisions: each time the life duration or the capacity of the vial is exceeded, a new vial is opened. Another version of this problem has been studied in [Billaut et al., 2011, Billaut, 2011].

Furthermore, because each preparation has to be delivered to a patient for a given due date, another objective of the problem is to minimize the maximum lateness related to these due dates.

The same type of problem arises in concrete production because once prepared, the concrete has to be used within a given amount of time. Similarly, in food production, after the food is out of the freezer, it has to be used within a given time limit. Another possible application of this study arises for the resources management in the cloud environment [Padhy and Patra, 2013]. For cloud providers, the problem is to allocate resources dynamically in the form of virtual machines to end users. Each task needs a virtual machine, i.e. a given quantity of RAM and of CPU. The tasks assigned to a resource cannot require more RAM and more CPU than the available quantity. Our problem, while different, presents also some similarities with parallel batch scheduling problems. But to the best of our knowledge, the problem that we consider in this paper has never been considered in the literature before.

The paper is organized as follows. In Section 2, the notations are introduced and the problem is formally defined. A MILP model is given. In Section 3, a recovering beam search and a matheuristic algorithm are proposed. These methods are tested on two-constraint bin packing benchmark instances with due dates considerations and without due dates for a comparison with methods dedicated to the classical two-constraint bin packing problem. The results presented in Section 4 show that the proposed methods have very good performances, even without due date considerations. In this latter case where 7 new bounds are obtained with respect to the available literature. Section 5 presents the conclusions and some future research directions.

2 Problem statement and notations

We consider a simplified version of the problem of chemotherapy drugs production, assuming that there is only one machine and only one type of product (raw material). We have a set of n jobs to schedule on a single machine. To each job $J_j \in \{1, ..., n\}$ is associated a processing time p_j, a

consumption b_j and a due date d_j. Without loss of generality, the jobs are supposed to be numbered in EDD order, i.e. $d_1 \leq d_2 \leq ... \leq d_n$. The life duration of the product after opening is equal to T and the volume of one vial is equal to V. We assume without loss of generality that $p_j < T$ and $b_j < V$, $\forall j$, $1 \leq j \leq n$. The number of vials is not limited but supposed to be bounded by n. We denote by C_j the completion time of J_j, L_j the lateness defined by $L_j = C_j - d_j$ and the maximum lateness is defined by $L_{\max} = \max_{1 \leq j \leq n} L_j$. We assume that the maximum lateness is bounded by a given value Q.

$$L_{\max} \leq Q \tag{1}$$

We also assume that the remaining quantity of product in the last opened vial is lost at the end of the time horizon. Therefore, minimizing the quantity of lost product is equivalent to minimizing the number of vials that are opened. That is the reason why the problem is a mixed between a scheduling problem and a two-constraint bin packing problem. Without due dates (or with extremely large due dates), the problem is exactly a two-constraint bin packing problem and thus clearly strongly NP-hard. With a big value of T and a big value of V, the problem is the classical single machine problem with the $L_{\max}$ minimization. In the following, we call a *bin*, the set of jobs performed with the same vial. It is well known from the literature that a trivial lower bound on the number of bins is the following

$$LB = \max\left(\left\lceil \sum_{j=1}^{n} p_j/T \right\rceil, \left\lceil \sum_{j=1}^{n} b_j/V \right\rceil\right). \tag{2}$$

We illustrate the problem by a numerical example.

Example

We consider a set of six jobs, $T = 10$ and $V = 10$.

j	1	2	3	4	5	6
p_j	3	4	4	5	3	1
b_j	1	2	5	3	1	4
d_j	7	9	11	13	14	16

Schedule $(J_1, J_3, J_5, J_2, J_4, J_6)$ is represented in Fig. 1. In this two dimensional Gantt chart, a job J_j is represented by a rectangle with the duration p_j on the x-axis and the consumption b_j on the y-axis. Jobs of the same bin are connected by the south-west corner of the rectangle.

The first job of a bin is put on the x-axis. In Fig. 1, one can see that job J_1 is the first job of the bin composed by jobs $\{J_1, J_3, J_5\}$ and job J_2 is the first job of the bin composed by jobs $\{J_2, J_4, J_6\}$. Job J_2 cannot be included in the first bin because the duration of this bin would exceed T. The maximum lateness of this sequence is equal to $L_{\max} = \max(-4, 5, -4, 6, -4, 4) = 6$ and this schedule requires two bins. Notice that schedule $(J_1, J_2, J_3, J_4, J_5, J_6)$ is optimal for the $L_{\max}$, but requires three bins (see Fig. 2).

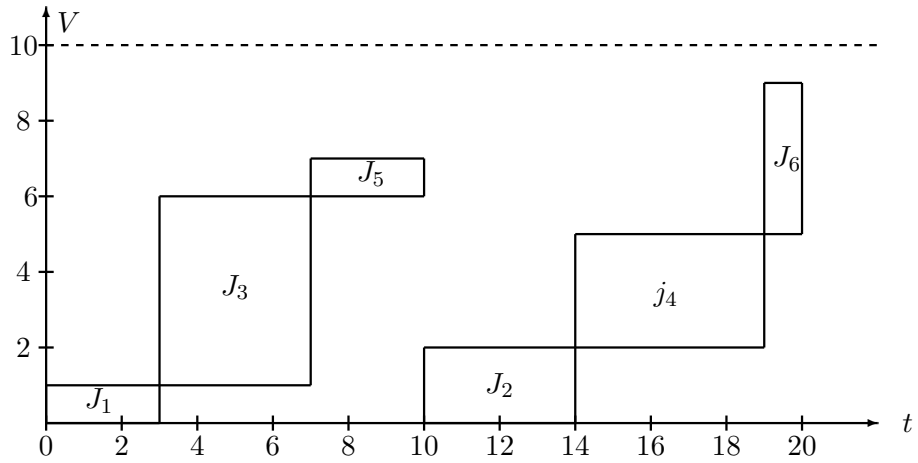

Figure 1: Two-dimensional Gantt chart representation of schedule $(J_1, J_3, J_5, J_2, J_4, J_6)$

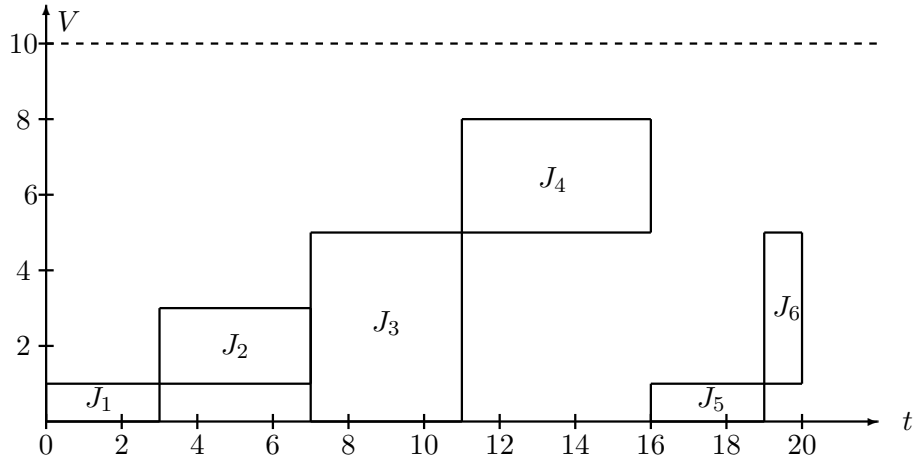

Figure 2: Two-dimensional Gantt chart representation of schedule $(J_1, J_2, J_3, J_4, J_5, J_6)$

Dominance condition

Proposition 1 *In a bin, the jobs are scheduled in EDD order.*

This condition is clear because the order in a bin does not modify the number of bins that are used, and tends to improve the $L_{\max}$ value.

MILP formulation

We now propose an MILP formulation of the problem.

We denote by $u_k \in \{0,1\}$ a boolean variable equal to 1 if bin k is used, and 0 otherwise and by $x_{j,k} \in \{0,1\}$ a boolean variable equal to 1 if job J_j is assigned to bin k, and 0 otherwise. The problem can be formalized by a binary linear program as follows.

$$\text{minimize } \sum_{k=1}^{n} u_k \tag{3}$$

subject to

$$\sum_{k=1}^{n} x_{j,k} = 1, \qquad \forall j \in \{1,...,n\} \tag{4}$$

$$\sum_{j=1}^{n} p_j x_{j,k} \leq T u_k, \qquad \forall k \in \{1,...,n\} \tag{5}$$

$$\sum_{j=1}^{n} b_j x_{j,k} \leq V u_k, \qquad \forall k \in \{1,...,n\} \tag{6}$$

$$\sum_{h=1}^{k-1} \sum_{i=1}^{n} p_i x_{i,h} + \sum_{i=1}^{j} p_i x_{i,k} \leq d_j + Q + M(1 - x_{j,k}), \quad \forall j \in \{1,...,n\}, \forall k \in \{1,...,n\} \tag{7}$$

$$u_{k+1} \leq u_k, \qquad \forall k \in \{1,...,n\} \tag{8}$$

Constraints (4) ensure that each job is performed by using one vial (is assigned to one bin). Constraints (5) and (6) correspond to the temporal and capacity limits. Constraints (7) suppose that job J_j is in bin k and correspond to the bound on the $L_{\max}$: $\sum_{h=1}^{k-1} \sum_{i=1}^{n} p_i x_{i,h}$ is the completion time of the $k-1^{th}$ bin, $\sum_{i=1}^{j} p_i x_{i,k}$ is the completion time of job J_j in its bin (remember that the jobs are numbered in EDD order). Constraints (8) ensure that the bins are used in their index increasing order. This model contains $n(n+1)$ binary variables and $n^2 + 4n$ constraints.

3 Resolution methods

Two original resolution methods are proposed for solving the problem. The first one is a recovering beam search algorithm and the second one is a matheuristic algorithm.

3.1 Recovering beam search algorithm

The *Beam Search* algorithm is a truncated branch-and-bound method where a subset of w nodes at each level are selected for branching. w is called the *beam width*. This method was first proposed in [Ow and Morton, 1988]. For the selection of nodes, each node is evaluated by a combination of a lower bound (LB) and an upper bound (UB), generally a weighted sum $WS = (1-\alpha)LB + \alpha UB$. Because the selected nodes are not necessarily the bests at a given level of the tree, among the set of possible nodes of a pure branch-and-bound algorithm, a recovering phase is applied in the *Recovering Beam Search* algorithm (RBS). The aim of this phase is to recover from wrong decisions jumping to a better node at the same level of the search tree. For a detailed description of RBS we refer to [Della Croce et al., 2004]. Fig. 3 illustrates the exploration of the tree in RBS with a beam width of 2.

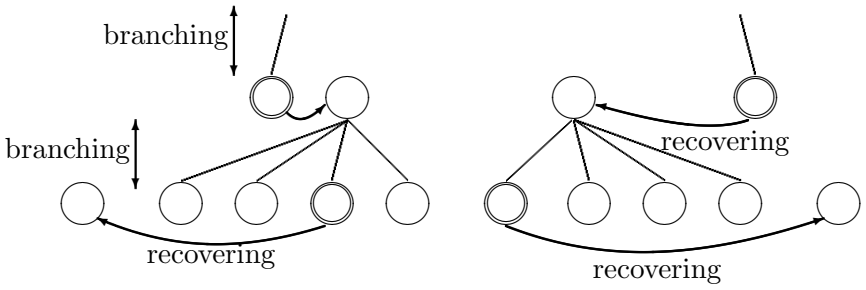

Figure 3: RBS exploration

The RBS method has already been successfully used in the literature for solving scheduling problems [Della Croce et al., 2004], [Dong et al., 2009], [Rakrouki et al., 2012], [Valente, 2010], [Esteve et al., 2006], [Ghirardi and Potts, 2005]. In order to apply the RBS approach, it is necessary to specify its main components, namely: branching scheme, lower bound, upper bound and recovering step. In this problem, a node of the search tree is defined by a partial sequence $S(\sigma)$ of the set σ of scheduled jobs, a set $\overline{\sigma}$ of unscheduled jobs, a lower bound $LB(\sigma)$, and an upper bound $UB(\sigma)$. At the root node, the initial sequence of unscheduled jobs is determined as follows. Starting from the EDD sequence, a steepest descent algorithm is used to reduce the number of bins, without violating the constraint on the L_{max}. The branching scheme is the typical n-ary branching: the sequence is constructed by adding one job at a time starting from position 1 and the search tree is such that a node at level k indicates which is the job placed in position k. For the lower bound computation, we denote by $Bin(S)$ a function that computes in $O(n)$ the number of bins used by a partial sequence S and the sum of jobs processing times and of jobs consumptions in the last bin (respectively called

$RestP(S)$ and $RestB(S)$). The following algorithm 1 provides the lower bound value computed at a generic node of the search tree.

Algorithm 1 $LB(\sigma)$

$NbBins = Bin(S(\sigma)) - 1$

$SumP = RestP(S(\sigma)) + \sum_{j \in \overline{\sigma}} p_j$

$SumB = RestB(S(\sigma)) + \sum_{j \in \overline{\sigma}} b_j$

$NbBins = NbBins + \max(\lceil SumP/T \rceil, \lceil SumB/V \rceil)$

Return($NbBins$)

Basically the lower bound is the trivial lower bound expressed by equation (2) updated in order to take into account the set of bins already filled by the partial sequence.

For the upper bound computation, let the following algorithm 2 provides the upper bound value computed at a generic node of the search tree where a partial sequence $S(\sigma)$ has already been defined. Let $S_{father}(\overline{\sigma})$ denote the sequence of the jobs in $\overline{\sigma}$ obtained by keeping for these jobs the same order they had in the upper bound of the father node in the search tree. Also, let $S(\sigma)//S_{father}(\overline{\sigma})$ denote the concatenation of subsequences $S(\sigma)$ and $S_{father}(\overline{\sigma})$.

Algorithm 2 $UB(\sigma)$

$S'(\sigma) = S(\sigma)//S_{father}(\overline{\sigma})$

If $S'(\sigma)$ is unfeasible (constraint on $L_{\max}$ violated), then $NbBins = \infty$

Else $NbBins = Bin(S'(\sigma))$

Return($NbBins$)

The rationale here is to keep for the jobs in $\overline{\sigma}$ the same order they had in the previous branch of the search tree. If the sequence provided by the upper bound computation is unfeasible, then, $UB(\sigma) = \infty$ and correspondingly the related search tree node is discarded. The evaluation of a node is given by $V(\sigma) = (1 - \alpha)LB(\sigma) + \alpha UB(\sigma)$. Preliminary testing indicated that best results were obtained with $\alpha = 0.2$. Typically, there is a huge number of solutions with the same value of upper and lower bounds. In order to make a difference between solutions having the same value at a level of the tree, a second-level evaluation has been used. Let us define the "*surface of a bin*", denoted by $Surf(k)$, as follows:

$$\text{Surf}(k) = \sum_{J_j \in B_k} p_j b_j$$

. Whenever two sequences have the same value of $V(\sigma)$, we break ties by selecting the sequence

with the smallest surface of the last bin.

The recovering phase is composed by two types of neighborhood called SWAP and EBSR (extraction and backward shift reinsertion) [Della Croce et al., 2004]. Let consider two jobs J_i and J_j in $S(\sigma) = \pi_1 J_i \pi_2 J_j \pi_3$, with J_i before J_j. SWAP generates sequence $\pi_1 J_j \pi_2 J_i \pi_3$ and EBSR generates sequence $\pi_1 J_j J_i \pi_2 \pi_3$.

3.2 Matheuristic algorithm

Matheuristics constitute a combination of exact methods and metaheuristics that exploit the strength of both approaches within a "hybrid" procedure. A distinguishing feature is the exploitation of nontrivial mathematical programming tools as part of the solution process. We refer to [Della Croce et al., 2013] for a general description of matheuristics.

Here, we propose a *Variable Partitioning Local Search* (VPLS) procedure which can be seen as a matheuristic neighborhood search approach based on a partial re-optimization of a variables partitioning. This approach can be also considered as a generalized $k - opt$ neigborhood search procedure. The VPLS framework can be seen as a local search approach for MIPs, expecially suited for $0-1$ variables, using a generalization of the k-exchange neighborhood.

Consider a general MIP formulation

$$\min c^T X \text{ s.t. } AX \leq b,\ X \in \{0,1\}$$

where $X^t = (x_1, x_2, .., x_n)$ is a vector of n variables of the problem and $\bar{X}^t = (\overline{x_1}, \overline{x_2}, .., \overline{x_n})$ is a feasible solution to the MIP. If this is the case, it is always possible to define a subset S of a defined size of variables indices $\{1, 2, .., n\}$. The neighborhood $N(\bar{X})$ consists of all solutions of the MIP where the j^{th} variable is equal to the value of the j^{th} variable in $\bar{X}$ for all $j \notin S$, namely $N(\bar{X}) = \{X \mid x_j = \overline{x_j},\ \forall j \notin S\}$. The resulting neighborhoods $N(\bar{X})$ can then be searched for an improving solution using a MIP-solver both optimally or approximately. The main idea stems in representing the MIP as a permutation problem where variables belonging to the current solution are partitioned into two sets. A first set S is then reoptimized by means of an MILP solver generating a permutated assignment, while variables in the second set $\bar{X} \setminus S$ keep the same assignment as in the current solution.

For the considered problem, the incumbent solution returned by the RBS algorithm induces correspondingly a sequence of the bins where we assume that γ bins are used. The neighborhood exploration works as follows. Starting with the first bin, consider the r-th bin in the sequence

along with bins $r+1$, $r+2$,..., $r+H-1$ with item set $S = S_r \cup S_{r+1} \cup ... \cup S_{r+H-1}$. Solve the problem of rescheduling the items in S so that $w(S_{r+H-1}) = \sum_{j \in S_{r+H-1}} w_j$ is minimized, where we use $w_j = \max\{b_j, p_j\}$ (this performance measure has been selected experimentally).

The rationale is to empty as much as possible bin $r+H-1$. If a (sub)sequence with $w(S_{r+H-1}) = 0$ is obtained, then one bin is saved, γ is reduced by one unit and the process can restart with $r = 1$. Alternatively, the new subsequence is kept anyway as the space of bin S_{r+H-1} has been optimized and will be used in the next iterate. The approach is then iterated for $r = 1, 2, ..., \gamma - H + 1$. Whenever $r = \gamma - H + 1$ is reached, the process restarts with $r = 1$ until a time limit is exceeded. The problem of rescheduling the items in S is done by solving by means of an ILP solver the following ILP model adapted from the one above in order to take into account the fact that bins $1, ...r-1$ and $r+H, ..., \gamma$ are not rescheduled and that the objective is to minimize the weight of the items assigned to bin $r+H-1$.

$$\text{minimize } z \tag{9}$$

subject to

$$\sum_{k=1}^{\gamma} x_{j,k} = 1, \qquad \forall j \in \{1, ..., n\} \tag{10}$$

$$\sum_{j=1}^{n} p_j x_{j,k} \leq T, \qquad \forall k \in \{1, ..., \gamma\} \tag{11}$$

$$\sum_{j=1}^{n} b_j x_{j,k} \leq V, \qquad \forall k \in \{1, ..., \gamma\} \tag{12}$$

$$\sum_{j=1}^{n} w_j x_{j,r\bar{k}} \leq z \qquad (\text{with } \bar{k} = r + H - 1) \tag{13}$$

$$x_{j,k} = \bar{x}_{j,k}, \quad \forall j \in \{1, ..., n\}, \forall k \in \{(1, ..., r-1) \cup (r+H, ..., \gamma)\} \tag{14}$$

$$\sum_{h=1}^{k-1} \sum_{i=1}^{n} p_i x_{i,h} + \sum_{i=1}^{j} p_i x_{i,k} \leq d_j + Q + M(1 - x_{j,k}), \qquad \forall j \in \{1, ..., n\}, \forall k \in \{1, ..., \gamma\} \tag{15}$$

with $w_j = \max\{p_j, b_j\}$, $\forall j$, $1 \leq j \leq n$.

Constraints (10) ensure that each job is performed by using one of the γ vials and correspond to constraints (4) of the previous model. Constraints (11)–(12) indicate the temporal and capacity limits of the γ vials and correspond to constraints (4)–(5) of the previous model. Constraints (13) link to the objective function z temporal and capacity consumption of vial $r+H-1$. Constraints

(14) indicate that the assignment of items to bins $1, ..., r-1$ and $r+H, ..., \gamma$ is kept as it is in the incumbent solution. Finally, (15) match constraints (7) of the previous model.

Figure 4 displays the matheuristic neighborhood search approach proposed.

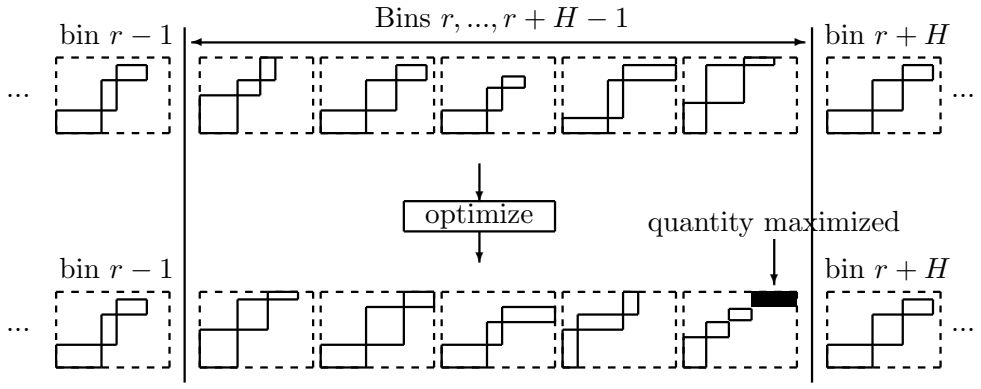

Figure 4: Illustration of the matheuristic algorithm

A pseudo-code of this procedure is depicted below.

Algorithm 3 SEARCH$(H, \gamma, \bar{x}, \bar{z})$

repeat

Set $r := 1$;

Set *improved*$[k]$:= **true** for each $k = 1, \ldots, \gamma$;

while $r \leq n - H + 1$ and $z^* > 0$ **do**

Solve model (9)–(14) with optimumz^*, x^*;

if $z^* = 0$ **then**

$\bar{x} := x^*$, $\bar{z} := z^*$;

$\gamma := \gamma - 1$;

end if

Set *improved*$[r + H - 1]$ = **true** iff $z^* < \sum_{j=1}^{n} w_j \bar{x}_{j,r+H-1}$;

Set $r := r + 1 \pmod{\gamma + H - 1}$;

end while

until not $\left(\bigvee_{k=1}^{\gamma} \text{improved}[k]\right)$ **or** ⟨time limit expired⟩;

4 Computational experiments

As there are no known instances for this problem, we present computational experiments on the set instances from the literature for the two-constraint bin packing problem to which we added correspondingly the jobs due dates. . The instances introduced by [Caprara and Toth, 2001] are grouped in 5 main classes, each class containing 10 instances for each value of n (30 instances per class in total with $n \in \{50, 100, 200\}$), which makes a total of 150 instances. These instances are available at:

`www.or.deis.unibo.it/research_pages/ORinstances/ORinstances.htm`

For each instance, the value in the first column is associated to the processing time and the value in the second column is associated to the vial consumption. Due dates have been randomly generated. We denote by P the sum of processing times: $P = \sum_{j=1}^{n} p_j$. For each job J_j, a due date is randomly generated in $[\alpha P(1-\beta), \alpha P(1+\beta)]$ with $\alpha \in [0,1]$ and $\beta \in [0,1]$ two coefficients. For the generation of the due dates, α and β have been fixed to 0.5.

For each data set, the EDD sequence has been computed for finding $L^*_{\max}$, the optimal value of the maximum lateness. For each instance, the bound Q is defined by $Q = \gamma L^*_{\max}$ with a given coefficient $\gamma \in \{1, 1000\}$. The value of $\gamma = 1$ imposes that the solution is optimal regarding the $L_{\max}$. On the other side, the large value for γ is equivalent to neglect the constraint on the $L_{\max}$ and the problem becomes equivalent to the original two-constraint bin packing problem. We denoted by RBS the results of the recovering beam search algorithm standalone and by "RBS-MH" the two-phase approach applying first RBS and then the matheuristic algorithm using the RBS solution as initial solution.

Table 1 reports the computational results for $\gamma = 1$. In this case, the solution has to be optimal for the maximum lateness. For each class and for each value of n, columns LB^* and UB^* of the table denote the best known lower and upper bounds respectively available for the two-constraint bin packing problem with no constraint on $L_{\max}$. These bounds were taken from [Monaci and Toth, 2006]. Also, column UB indicates the sum of the number of bins obtained for the 10 instances by the related method, column CPU(s) for RBS indicates the average computation time and column Ttb(s) for RBS-MH indicates the average time for obtaining the best solution (time-to-best). Note that for RBS a time limit of 120 seconds has been imposed while for the matheuristic algorithm a time limit of 1800 has been imposed, so that globally RBS-MH runs at most for 1920 seconds. Enlarging such limit does not seem to improve substantially the solution

quality. Note also that for each iteration of the matheuristic algorithm a time limit of 60 seconds is imposed.

$\gamma = 1$				RBS		RBS-MH	
Class	n	LB^*	UB^*	UB	CPU(s)	UB	Ttb(s)
1	50	135	135	144	0.9	136	1.3
	100	255	260	287	25.8	260	10.8
	200	503	510	571	120	511	98.0
6	50	214	215	242	0.8	221	2.2
	100	405	410	483	11.6	426	40.5
	200	803	811	988	120	821	437.8
7	50	196	197	219	0.9	203	1.2
	100	398	405	444	17.6	412	5.0
	200	799	802	927	120	808	170.9
9	50	144	145	160	0.97	147	1.2
	100	257	267	297	18.8	268	15.7
	200	503	513	582	120	514	235.4
10	50	170	170	207	1.1	185	5.6
	100	330	330	396	16.9	350	20.9
	200	670	670	840	120	680	277.7

Table 1: Results of the algorithms for the two-constraint bin packing instances with due dates ($\gamma = 1$)

From the results, if we compare the values of the proposed approaches to the best feasible solutions (indicated by the entry UB^*) known for the two-constraint bin packing problem, we notice that the solution quality of the combined RBS-MH method is particularly good on classes $1, 9$, still well performing on class 7 and with partial degradation on classes $6, 10$. Also, the time-to-best is typically significantly inferior to the 600 seconds of CPU time. Notice, however, that the strong constraint on $L_{\max}$ does not necessarily guarantee that the UB^* values are reachable. The performance of RBS standalone are weaker, but require limited time. Notice also, that additional tests not reported here indicate that the performances of RBS are much better than the standard list scheduling EDD (earliest due date) sequence.

4.1 Instances without due dates

We consider in this subsection the case $\gamma = 1000$. In this case, due dates are not considered and hence the original bin packing instances are solved. Correspondingly, constraint (15) becomes redundant and the bins selected for rescheduling are no more required to be consecutive. Hence, the following simpler ILP model can be considered in the matheuristic.

$$\text{minimize } z \tag{16}$$

subject to (10), (11), (12), (14) and

$$\sum_{j=1}^{n} w_j x_{j,\bar{k}} \leq z \qquad \text{for a chosen } \bar{k} \in I \tag{17}$$

Also, a more general matheuristic approach could be devised as no sequencing constraint are present anymore. The approach works as follows. First, H disjoint subsets of bins with H bins in each subset are selected and optimized in order to empty as much as possible one bin in each subset. Then the H disjoint emptied bins are re-optimized always trying to empty as much as possible one of them. The approach is iterated until a time limit expires. A pseudo-code of this approach is depicted below.

Algorithm 4 SEARCH-BP$(H, \gamma, \bar{x}, \bar{z})$

repeat

$N = \{1, \ldots, \gamma\}$;

Randomly select H disjoint index sets $I_1, \ldots, I_H \subset N$ with $|I_1| = \cdots = I_H = H$;

for each I_r, $r = 1, \ldots, H$ **do**

Solve model (16)–(17) for I_r, with optimum z^*, x^*;

end for

Let $I' = \{k\colon k = \arg\min\{\sum_{j=1}^{n} w_j x^*_{j,k},\ k \in I_r,\ r = 1, \ldots, H\}\}$;

Solve model (16)–(17) for I', with optimum z', x';

if $z' = 0$ **then**

Set $\gamma := \gamma - 1$;

end if

until ⟨time limit expired⟩;

Table 2 provides the same entries of Table 1 for the original two-constraint bin packing problem that is the case with no constraints on the due dates. The results show that RBS-MH is competitive

with the state of the art procedures being superior on class 1, comparable on classes $7, 9$ and inferior on classes $6, 10$. We point out, in particular, that as far as class 1 is concerned, 2 new upper bounds were established for $n = 100$ and 5 new upper bounds were established for $n = 200$. The corresponding solutions are available at ... Also, the CPU time required by RBS-MH is on the average is wee below 400 seconds.

$\gamma = 1000$				RBS		RBS-MH	
Class	n	LB^*	UB^*	UB	CPU(s)	UB	Ttb(s)
1	50	135	135	139	3.7	135	0.1
	100	255	260	263	98.4	258	14.3
	200	503	510	527	120	505	190.9
6	50	214	215	219	3.0	215	0.5
	100	405	410	421	30.6	410	48.1
	200	803	811	847	120	819	233.2
7	50	196	197	201	3.8	198	1.1
	100	398	405	406	23.7	405	0.1
	200	799	802	810	120	803	134.5
9	50	144	145	146	2.7	145	0.1
	100	257	267	274	48.7	267	0.5
	200	503	513	533	120	513	7.3
10	50	170	170	187	4.2	180	0.4
	100	330	330	351	28.4	340	2.5
	200	670	670	713	120	680	30.7

Table 2: Results of the algorithms for the two-constraint bin packing instances without due dates ($\gamma = 1000$)

5 Conclusions

We considered the problem of scheduling activities which consume perishable raw materials that induce a deadline on their use once started where the deadline is a part of the decision process. The problem has been modeled as a single machine scheduling problem with additional duration and consumption constraints. The problem can also be seen as a two-constraint bin packing problem

combined to the single machine problem with the requirement that the maximum lateness of the resulting single machine sequence does not exceed a given threshold. A two-phase procedure has been proposed where in the first phase a fast Recovering Beam Search approach has been applied and in the second phase a more time consuming matheuristic procedure using the RBS solution as initial solution has been proposed. The two procedures have been tested showing very good performances within reasonable time on benchmark two-constraint bin packing instances suitably integrated to handle the maximum lateness requirement. The proposed procedures have also been tested on the original two-constraint bin packing instances showing to be competitive with the state of the art procedures and finding 8 new best bounds. As a future research direction, it would be worthy to adapt the proposed approach to the perishable raw material version of the original problem discussed in [Mazier et al., 2010].

References

[Billaut et al., 2011] J-C. Billaut, P. Esquirol, J-F. Tournamille (2011). Scheduling chemotherapy preparations with perishable raw material constraints. Informs Healthcare, Montreal, Canada, June 20–22.

[Billaut, 2011] J-C. Billaut (2011). New scheduling problems with perishable raw materials constraints. 16th IEEE IInternational conference on Emerging Technologies and Factory Automation, Toulouse, France, September 5–9.

[Brucker, 2007] P. Brucker (2007). Scheduling algorithms. Fifth edition. Springer Berlin, Heidelberg, New York.

[Caprara and Toth, 2001] A. Caprara, P. Toth (2001). Lower bounds and algorithms for the 2-dimensional vector packing problem. *Discrete Applied Mathematics*, 111:231–262.

[De Carvalho, 2002] Valério De Carvalho, J.M. (2002). LP models for bin packing and cutting stock problems *European Journal of Operational Research*, 141 (2), pp. 253-273.

[Della Croce et al., 2004] F. Della Croce, M. Ghirardi, R. Tadei (2004). Recovering Beam Search: enhancing the beam search approach for combinatorial optimization problems. *Journal of Heuristics*, 10:89–104.

[Della Croce et al., 2013] F. Della Croce, A. Grosso, F. Salassa (2013). Matheuristics: embedding MILP solvers into heuristic algorithms for combinatorial optimization problems. In "Heuristics : Theory and Applications", P. Siarry ed., Nova Science Publishers, pp. 31-52.

[Dong et al., 2009] X. Dong, H. Huang, P. Chen (2009). An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion. *Computers & Operations Research*, 36:1664–1669.

[Esteve et al., 2006] B. Esteve, C. Aubijoux, A. Chartier, V. T'kindt (2006). A recovering beam search algorithm for the single machine Just-in-Time scheduling problem. *European Journal of Operational Research*, 172(3): 798–813.

[Ghirardi and Potts, 2005] M. Ghirardi, C.N. Potts (2005). Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach. *European Journal of Operational Research*, 165(2): 457–467.

[Hartman and Briskorn, 2010] S. Hartmann, D. Briskorn (2010). A survey of variants and extensions of the resource-constrained project scheduling problem *European Journal of Operational Research*, 207 (1), pp. 1–14.

[James and Almada-Lobo, 2011] R.J.W. James, B. Almada-Lobo (2011). Single and parallel machine capacitated lotsizing and scheduling: New iterative MIP-based neighborhood search heuristics. *Computers and Operations Research*, 38 (12), pp. 1816–1825.

[Lodi et al., 2002] A. Lodi, S. Martello, D. Vigo (2002). Recent advances on two-dimensional bin packing problems *Discrete Applied Mathematics*, 123 (1-3), pp. 379–396.

[Mazier et al., 2010] A. Mazier, J-C. Billaut, J-F. Tournamille (2010). Scheduling preparation of doses for a chemotherapy service. *Annals of Operations Research*, 178(1): 145–154.

[Monaci and Toth, 2006] M. Monaci, P. Toth (2006). A Set-Covering-Based Heuristic Approach for Bin-Packing Problems. *INFORMS Journal on Computing*, 18 (1): 71–85.

[Ow and Morton, 1988] P. S. Ow, T. E. Morton (1988). Filtered beam search in scheduling. *International Journal of Production Research*, 26: 297–307.

[Padhy and Patra, 2013] R. P. Padhy, M. R. Patra (2013). Service support aware resource allocation policy for enterprise cloud-based systems. *International Journal of Cloud Computing and Services Science*, 2(4): 296–312.

[Pinedo, 2012] M. Pinedo (2012). Scheduling. Theory, algorithms and systems. Fourth edition. Springer.

[Rakrouki et al., 2012] M.A. Rakrouki, T. Ladhari, V. T'Kindt (2012). Coupling Genetic Local Search and Recovering Beam Search algorithms for minimizing the total completion time in the single machine scheduling problem subject to release dates. *Computers and Operations Research*, 39(6): 1257–1264.

[Ruiz and Vázquez-Rodrguez, 2010] R. Ruiz, J.A. Vázquez-Rodrguez (2010). The hybrid flow shop scheduling problem *European Journal of Operational Research*, 205 (1), pp. 1–18.

[Valente, 2010] J.M.S. Valente (2010). Beam search heuristics for quadratic earliness and tardiness scheduling. *Journal of the Operational Research Society*, 61(4): 620–631.